NVM Express Technical Errata

Errata ID	**015**
Change Date	**6/15/2011**
Affected Spec Ver.	**NVM Express 1.0a**
Corrected Spec Ver.	

Submission info

Name	Company	Date
Kevin Marks	Dell	6/15/2011
Peter Onufryk	IDT	6/15/2011

This erratum makes clarifications and corrections to the end-to-end data protection material.

This erratum makes clarifications to the Firmware Image Download and Get Log Page commands.

Description of the specification technical flaw

Update the last paragraph of section 8.2 as shown (includes ECN 013 updates):

If end-to-end data protection is used, then the Protection Information field ~~eight bytes of metadata~~ for each logical block is ~~used for the Protection Information field~~ contained in the metadata.

Update the first four paragraphs of section 8.3 as shown below:

To provide robust data protection from the application to the NVM media and back to the application itself, end-to-end data protection may be used. If ~~When~~ this optional mechanism is enabled, then additional protection information (e.g. CRC) is added to the ~~LBA~~ logical block that may be evaluated by the controller and/or host software to determine the integrity of the ~~LBA~~ logical block. This additional protection information, if present, is either the first eight bytes of metadata or the last eight bytes of metadata, based on the format of the namespace. For metadata formats with more than eight bytes, if the protection information is contained within the first eight bytes of metadata, then the CRC does not cover any metadata bytes. For metadata formats with more than eight bytes, if the protection information is contained within the last eight bytes of metadata, then the CRC covers all metadata bytes up to but excluding these last eight bytes. As described in section 8.2, metadata and hence this protection information may be configured to be contiguous with the ~~LBA~~ logical block data or stored in a separate buffer.

The most commonly used data protection mechanisms in Enterprise implementations are Data Integrity Field (DIF), defined in the SCSI Block Commands – 3 reference (SBC-3), and the Data Integrity Extension (DIX). The primary difference between these two mechanisms is the location of the protection information. In DIF the protection information is contiguous with the ~~LBA~~ logical block data and creates an extended ~~LBA~~ logical block, while in DIX the protection information is stored in a separate buffer. The end-to-end data protection mechanism defined by this specification is functionally compatible with both ~~T10~~ DIF and DIX. DIF functionality is achieved by configuring the metadata to be contiguous with ~~LBA~~ logical block data (as shown in Figure 139, while DIX functionality is achieved by configuring the metadata and data to be in separate buffers (as shown in Figure 140).

NVM Express supports the same end-to-end protection types as ~~T10~~ DIF. The type of end-to-end data protection (Type 1, Type 2, or Type 3) is selected when a namespace is formatted and is reported in the Identify Namespace data structure.

The Protection Information format is shown in Figure 141 and is contained in the ~~eight bytes of~~ metadata associated with each ~~LBA~~ logical block. The Guard field contains a CRC-16 computed over the ~~LBA~~ logical block data. In addition to a CRC-16, DIX also specifies an optional IP checksum that is not supported by NVM Express. The Application Tag is an opaque data field not interpreted by the controller and that may be used to disable checking of protection information. The Reference Tag associates ~~LBA~~ logical block data with an address and protects against misdirected or out-of-order ~~LBA~~ logical block transfer. Like the Application Tag, the Reference Tag may also be used to disable checking of protection information.

Update the fifth paragraph of section 8.3 as shown below:

The protection information processing performed as a side effect of Read and Write commands is controlled by the Protection Information Action (PRACT) bit in the command. Figure 142 illustrates the protection information processing that may occur as a side effect of a Write command. If the namespace is not formatted with end-to-end data protection, then ~~LBA~~ logical block data and any optional metadata is transferred from the host to the NVM with no processing. If the namespace was formatted with protection information and the PRACT bit is cleared to ‘0’, then ~~LBA~~ logical block data and metadata containing protection information are transferred from the host to NVM. As the ~~LBA~~ logical block and metadata passes through the controller, the protection information is checked. If a protection information check error is detected, the command completes with the status code of the error detected (i.e., End-to-end Guard Check, End-to-end Application Tag Check or End-to-end Reference Tag Check). If the namespace was formatted with protection information and the PRACT bit is set to ‘1’, then ~~LBA~~ logical block data is transferred from the host to the controller. The controller inserts protection information and the ~~LBA~~ logical block data and metadata containing protection information are written to NVM.

Update the paragraph preceding Figure 143 in section 8.3 as shown below:

Figure 143 illustrates the protection information processing that may occur as a side effect of Read command processing. The processing parallels Write command processing with the exception that ~~LBA~~ logical block data flows in the opposite direction. When the PRACT bit is cleared to ‘0’ and the namespace was formatted with protection information, ~~LBA~~ logical block data and metadata are transferred from NVM to the host and checked by the controller. When the PRACT bit is set to ‘1’ and the namespace was formatted with protection information, ~~LBA~~ logical block data and metadata are transferred from the NVM to the controller. The controller checks the protection information and then removes it from the metadata before passing the LBA to the host. If the namespace format contains metadata beyond the protection information, then the protection information is not stripped regardless of the state of the PRACT bit (i.e., the metadata field remains the same size in the host as that in NVM).

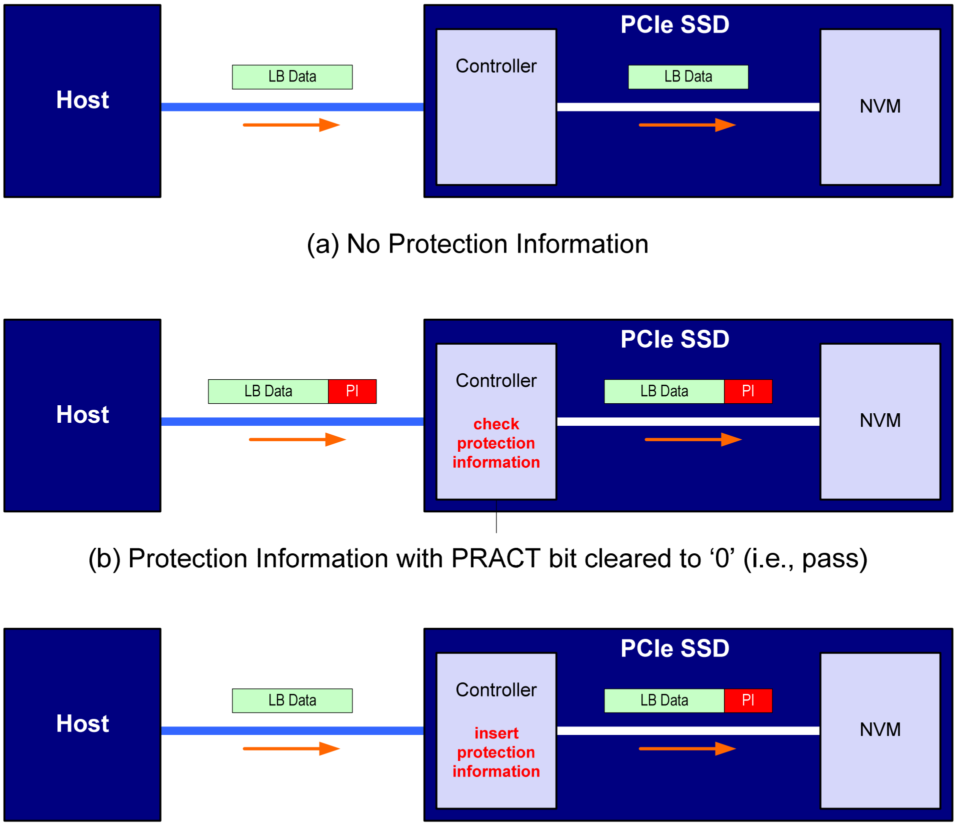
Update the three paragraphs following Figure 143 in section 8.3 as shown below:

Protection processing for fused operations is the same as those for the individual commands that make up the fused operation.

Checking of protection information consists of the following operations performed by the controller. If bit ~~28~~ 2 of the Protection Information Check (PRCHK) field of the command is set to ‘1’, then the controller compares the protection information Guard field to the CRC-16 computed over the ~~LBA~~ logical block data. If bit ~~27~~ 1 of the PRCHK field is set to ‘1’, then the controller compares unmasked bits in the protection information Application Tag field to the Logical Block Application Tag (LBAT) field in the command. A bit in the protection information Application Tag field is masked if the corresponding bit is cleared to ‘0’ in the Logical Block Application Tag Mask (LBATM) field of the command. If bit ~~26~~ 0 of the PRCHK field is set to ‘1’, then the controller compares the protection information Reference Tag field to the computed reference tag. The value of the computed reference tag for the first LBA of the command is the value contained in ~~of~~ the Initial Logical Block Reference Tag (ILBRT) field in the command. The computed reference tag is incremented for each subsequent ~~LBA~~ logical block. Unlike ~~T10~~ DIF Type 1 protection which implicitly uses the least significant four bytes of the LBA, The controller always uses the ILBRT field and requires host software to initialize the ILBRT field to the least significant four bytes of the LBA when Type 1 protection is used.

Protection checking may be disabled as a side effect of the value of the protection information Application Tag and Reference Tag fields regardless of the state of the PRCHK field in the command. ~~When~~ If the namespace is formatted for Type 1 or Type 2 protection, then all protection information checks are disabled regardless of the state of the PRCHK field when the protection information Application Tag has a value of FFFFh. ~~When~~ If the namespace is formatted for Type 3 protection, then all protection information checks are disabled regardless of the state of the PRCHK field when the protection information Application Tag has a value of FFFFh and the protection information Reference Tag has a value of FFFF_FFFFh.

Replace Figure 142 with updated figure below. LBA Data changed to LB Data and description for item b) and c) updated.

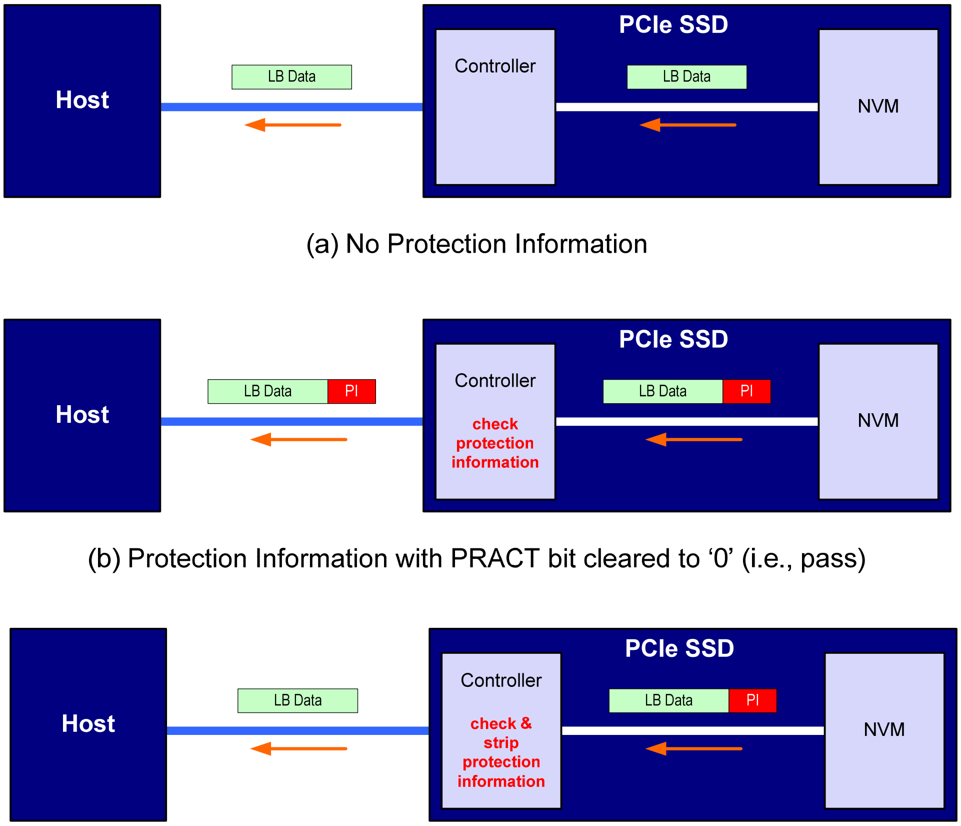


(a) No Protection Information

(b) Protection Information with PRACT bit cleared to ‘0’ (i.e., pass)

(c) Protection Information with PRACT bit set to ‘1’ (i.e., insert)

Replace Figure 143 with updated figure below. LBA Data changed to LB Data and description for item b) and c) updated.



(a) No Protection Information

(b) Protection Information with PRACT bit cleared to ‘0’ (i.e., pass)

(c) Protection Information with PRACT bit set to ‘1’ (i.e., strip)

Modify the second paragraph of section 5.8 as shown below:

The firmware image may be constructed of multiple pieces that are individually downloaded with separate Firmware Image Download commands. Each Firmware Image Download command includes a Dword Offset and Number of Dwords that ~~indicates~~ specify a Dword range. The host software shall ensure that firmware pieces do not have Dword ranges that overlap. Firmware portions may be issued out of order to the controller.

The new firmware image is not applied as part of the Firmware Image Download command. It is applied following a reset, where the image to apply and the firmware slot it should be committed to is ~~indicated~~ specified with the Firmware Activate command.

Modify section 5.8.1 as shown below:

A completion queue entry is posted to the Admin Completion Queue ~~when~~ if this portion of the firmware image has been ~~stored~~ received by the controller.

Modify Figures 46-48 as shown below:

Figure 46: Firmware Image Download – PRP Entry 1

Bit	Description
63:00	**PRP Entry 1 (PRP1):** This field contains the first PRP entry, ~~indicating~~ specifying the location where data should be transferred from.

Figure 47: Firmware Image Download – PRP Entry 2

Bit	Description
63:00	**PRP Entry 2 (PRP2):** This field contains the second PRP entry. If the data transfer is satisfied with PRP Entry 1, then this field is reserved. If the data transfer may be satisifed with two PRP entries total, then this entry specifies the location where data should be transferred from. If the data transfer requires more than two PRP entries, then this field ~~includes~~ contains a pointer to a PRP List.

Figure 48: Firmware Image Download – Command Dword 10

Bit	Description
31:00	**Number of Dwords (NUMD):** This field ~~indicates~~ specifies the number of Dwords to transfer for this portion of the firmware. This is a 0's based value.

Modify the first paragraph of section 5.8.1 as shown below:

The Get Log Page command returns a data buffer ~~that contains~~ containing the log page requested.

Update Figure 54 as shown below:

Figure 54: Get Log Page – PRP Entry 1

Bit	Description
63:00	**PRP Entry 1 (PRP1):** ~~Indicates~~ Specifies a data buffer that the log page shall be returned to. The buffer shall not have more than one physical discontinuity and shall be 4KB minimum in size.

Update Figure 56 as shown below:

Figure 56: Get Log Page – Command Dword 10

Bit	Description
31:28	Reserved
27:16	**Number of Dwords (NUMD):** This field specifies the number of Dwords to return. If host software ~~indicates~~ specifies a size larger than the log page requested, the results are undefined. This is a 0's based value.
15:08	Reserved
~~15:00~~ 07:00	**Log Page Identifier (LID):** This field specifies the identifier of the log page to retrieve.

Disposition log

6/15/2011	Erratum captured.
8/1/2011	Erratum ratified.

Technical input submitted to the NVMHCI Workgroup is subject to the terms of the NVMHCI Contributor's agreement.